Radio-hosted Flight / Ground Interface for Operations Standardization

Dr. Christopher A. Grasso[1]
Blue Sun Enterprises, Boulder, Colorado, 80302
and
Robert Lock[2] and Patricia d. Lock[3]
Jet Propulsion Laboratory / California Institute of Technology, Pasadena, California, 91109

The interface between a spacecraft and its ground operations segment includes the flow of commands, configuration, and sequencing elements to the spacecraft, and the flow of telemetry and data products from the spacecraft. Creating and implementing a complete definition of this interface simplifies and standardizes mission operations, allowing easy sharing of operations personnel across missions. Early spacecraft featured a simple flight / ground interface (FGI) using hardware command decoding in the radio, driven by technological limitations of the time. Modern spacecraft use command and data handling (CDH) avionics on which flight software executes, which in turn controls and configures the mission, executes subsystem and instrument instructions, and implements critical fault protection actions. Deep space missions feature advanced operations software for running sequenced activities over a period of weeks, which allows them to function with only infrequent ground contact. This approach comes at the cost of increased complexity in the FGI, requiring expensive modifications to heritage flight software and ground systems. By hosting the interface in the radio instead of the CDH avionics, modern missions can approximate the FGI design simplicity of early spacecraft, with significant advantages for vendor competition, lowered costs, standardization of operations, and reduction of implementation risk.

I. Mission Operations Domain

Mission operations for spacecraft involves both the uplink of products to the spacecraft and the return of mission data. Deep space mission operations may be viewed as a function of the mission objectives, and consists of three items, in priority order:

1. Collect science data to achieve the mission objectives

---

[1] Principal VML Engineer, Blue Sun Enterprises, 1942 Broadway Suite 314, Boulder, CO, 80302, Senior Member.

[2] Orbiter Concept Development Manager, Mars Program Formulation Office, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA, 91109, Member.

[3] Mission Operations Assurance Manager, Office of Safety and Mission Assurance, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA, 91109, Member.

2. Operate the flight system to collect science
3. Build, test, and deliver the flight system in order to have a platform to operate

Collecting science to achieve the mission objectives (objective 1) requires successful operation of the flight system to collect the science (objective 2). The more operable the flight system design (objective 3), the simpler the operations team's job becomes. Operability therefore has a substantial impact on cost, risk, and data return [14].

Products radiated include, at a minimum, immediate commands that are interpreted by the spacecraft in order to perform an action. Deep space missions are subject to light speed delays that drive the need for autonomy. Therefore, those missions generally employ stored commands called *sequences*. Mission operations products are defined in the context of a Mission Operations System (MOS). The MOS includes a Ground System (GS) that assists users in deriving the products needed onboard the spacecraft, as shown in Figure 1. Products to be delivered to the spacecraft include commands, sequences, configuration information, and software patches.

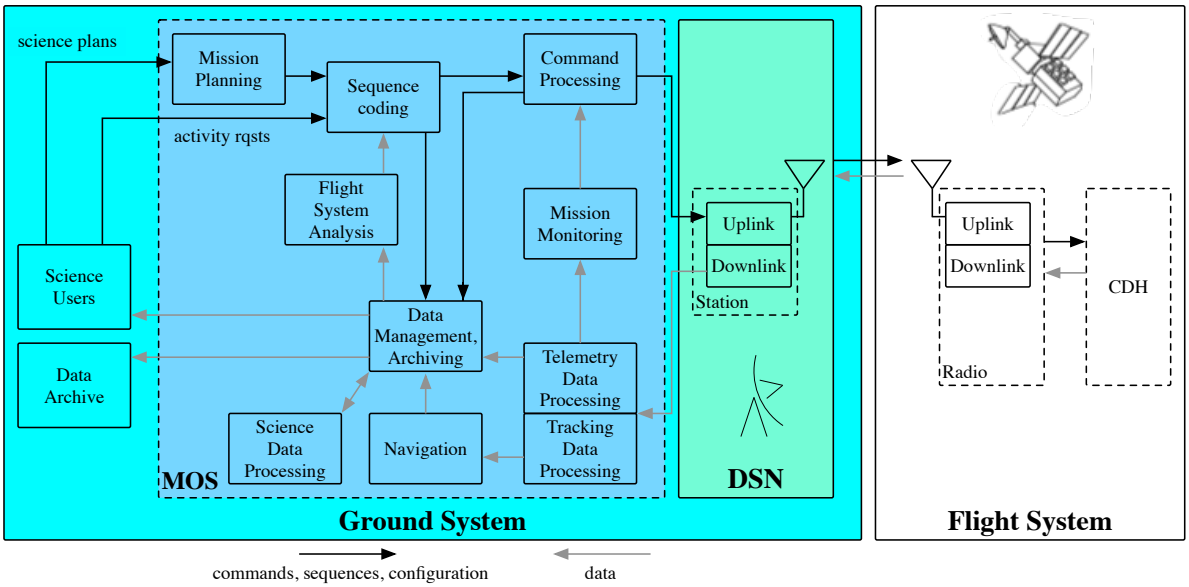

Figure 1: A simplified view of mission operations functions, showing the flow of data through the system

Onboard the spacecraft, the GS interfaces to the command system and file system of the spacecraft in order to deliver products used onboard to produce science. Sequences execute commands from an onboard store based on time and, in some cases, events and conditions. The sequence store frequently takes the form of files to be loaded, as do configuration data which govern the low-level operation of various software tasks and hardware elements. By migrating some portions of the interface to the radio onboard, we will show that the adaptations required to the MOS and spacecraft in order to integrate them together can be virtually eliminated.

II. Predecessor concepts

A. Hardware commands

A *hardware command decoder* translates commands received by the radio directly into actions aboard the spacecraft. Commandable spacecraft originally featured only hardware commands, wherein the bit stream arriving in the radio was tested by the hardware command decoder, and all spacecraft actuation was directly implemented: typically, the opening or closing

of switches and the setting of signals as shown in Figure 2 [13]. This uses the radio as the command subsystem, in essence hosting within the radio the flight-ground interface for control of the spacecraft. Commercially available radios had well-defined signal interfaces that could be connected to flight hardware, thereby standardizing the flight-ground interface on spacecraft using the same radio.

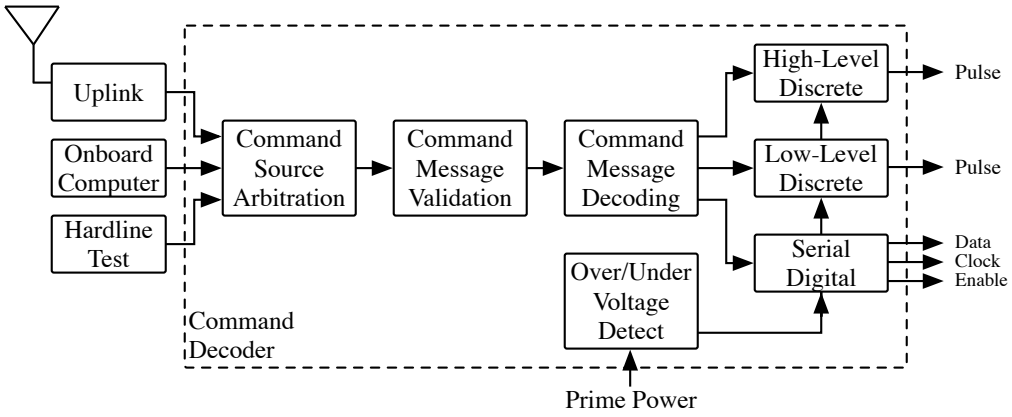

Figure 2: Command path from radio to actuators through hardware command decoder

Modern spacecraft generally reserve hardware commanding for critical spacecraft instructions to be executed in the event that flight software is no longer responsive. These critical commands may include resetting components like the CDH or radio, selecting hardware sides where redundant hardware is available, or modifying critical memory registers.

B. Modern flight software architecture with software commands

On modern spacecraft, the presence of a command and data handling (CDH) unit running flight software typically dispatches the majority of commands, using software interpretation of standardized command packets (e.g. CCSDS packets). By centralizing the command interpretation within the flight software, commands of considerable complexity may be implemented, to which the flight-ground system must be adapted.

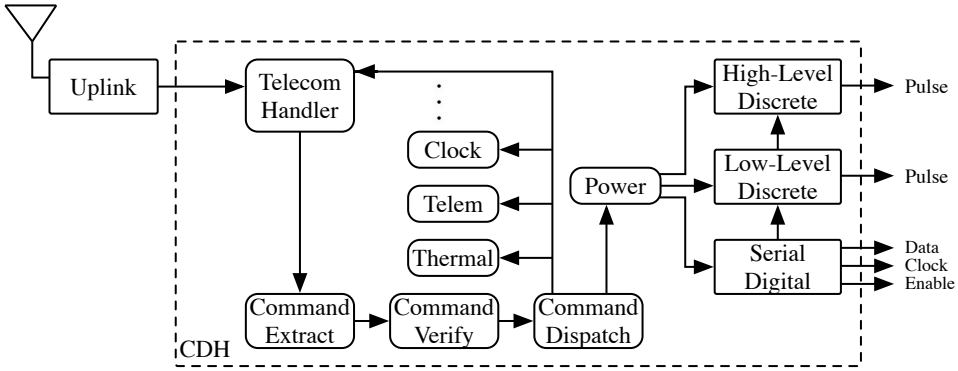

Figure 3: Command path from radio to software

Figure 3 illustrates the data flow paths of a CCSDS command packet from uplink, the extraction of data, and the dispatch of this data by the flight software to the appropriate software component for execution. Flight software executing on the CDH obviates the need for an extensive set of hardware commands, instead utilizing a software command dispatch system.

C. Virtual Machine Language sequencing

Sequencing is the execution of commands from a store onboard the spacecraft. Spacecraft featuring simple sequencing execute commands according to *absolute time*, i.e. when a specific time comes to pass. More modern sequencing typically also allows *relative time* sequencing, wherein the execution of a statement is dependent on the completion of a prior statement rather than an absolute time value. Modern sequencing systems may also allow reusable sequence elements, conditional execution, and event reaction.

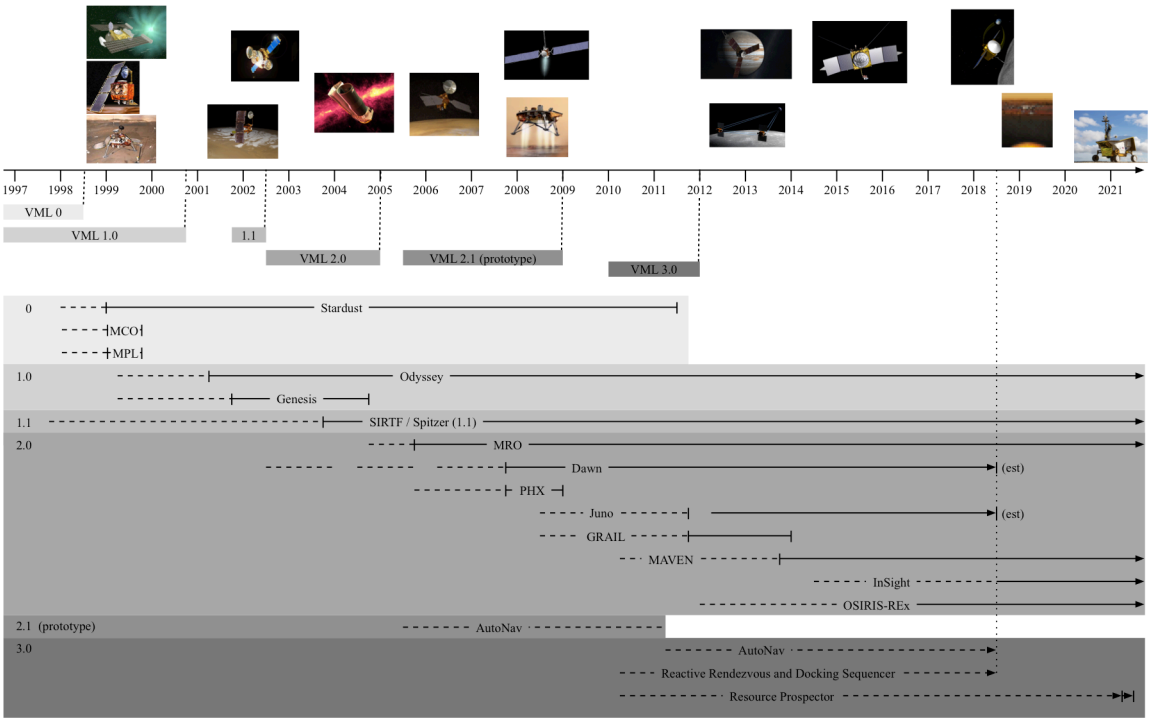

Figure 4: VML heritage, 1997 - 2018, 88 flight years, showing VML versions used on a variety of NASA deep space missions, from original VML 0 through commercialized VML 3.0

Virtual Machine Language (VML) is an advanced sequence processing language specifically tuned to the needs of spacecraft operations. It contains sufficient functionality to allow operators to implement solutions that in the past would have required the development of expensive, mission-unique flight software. The language is simple enough that it avoids most of the problems associated with typical flight software developed in C, C++, or Ada, while providing enough flexibility to implement elegant, straightforward operations.

VML does not run on the "bare iron" of the host microprocessor. Instead, the language is implemented as a byte code binary, and is interpreted at runtime by onboard software known as the VML Flight Component. This approach provides a safe sandbox for execution, eliminating issues such as race conditions, out-of-bounds memory accesses, division by zero, type coercion errors, or missing functions. It also simplifies deployment to a wide variety of processors.

Virtual Machine Language development started in 1997. Six versions have been implemented so far. VML has been used or is in use on fifteen NASA flight missions and technology demonstrators to date, including Stardust[2], Genesis, Mars Odyssey, Spitzer Space Telescope[3][4], MRO, Dawn, Phoenix[11][12], Juno, GRAIL, MAVEN, OSIRIS-REx, InSight, and the Resource Prospector lunar regolith analysis instrument package. A timeline of software development and use over the last 20 years appears in Figure 4.

The VML flight execution environment [1] provides multiple threads of parallel execution within one operating system task context using a data-driven construct known as a *sequencing engine*. VML allows an extensive set of variable types, including integers, floats, Boolean values, strings, and arrays. Arithmetic and trigonometric calculations, logical manipulations, and vector/matrix operations are available for use. Conditionals may also be used to make decisions based on local values at runtime. WHILE and FOR loops perform iteration.

VML sequences exist as named functions which can accept parameters and have local variables. Functions may be packaged together into a single file that is loaded onto an engine in order to associate runtime behavior or to provide libraries of commonly needed services. *Objects* with methods package code and data together, simplifying development and management of products. Specialized objects called *state machines* provide a directly-executable set of reactive actions, and can intrinsically coordinate together to perform sophisticated autonomy as an expert system.

D. Spacecraft Telecommand Radio System: STRS

Modern space radios are implemented in a software-defined fashion rather than as specialized hardware, using a CPU and software running at sufficient speed to perform processing on arriving signals in order to encode and decode data and manage hardware. Because of the presence of a processor and memory, these software-defined radios can host flight software of sufficient complexity to operate a mission. NASA's Spacecraft Telecommand Radio System (STRS) [15] is one set of standardized software specifications for such a radio, finding wide application in the arena of spacecraft communications. STRS explicitly provides for application hosting within the radio, making it an excellent candidate environment for executing sequencing, commanding, and specialized navigation needed for complex deep space missions. The STRS system is illustrated in Figure 5.

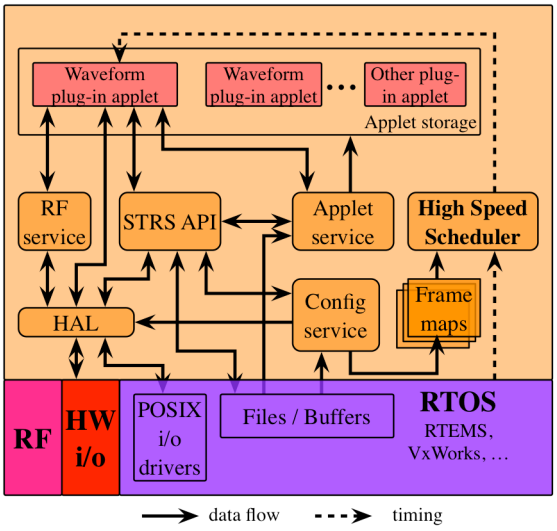

Figure 5: STRS software architecture controlling radio and hosting waveform applications

STRS features a well-defined application programming interface accessible to a series of waveform applications which can be dynamically loaded and unloaded as needed at runtime, allowing the radio to be updated in-flight with modified and new capabilities. Applications interface to the radio hardware through a hardware abstraction layer (HAL), which in turn interfaces to the hardware either directly or via POSIX-compliant input/output drivers. STRS runs atop a real-time operating system like RTEMS[16].

III. Using the radio as an extension to the flight/ground interface

Currently, a multi-mission MOS must incorporate a new spacecraft, the team can:

1) Modify the MOS to accommodate a new and potentially very different command and sequencing paradigm, with associated cost and schedule risk
2) Modify the flight software of the new spacecraft with a known sequencing capability, with associated cost, schedule risk, and performance risk

Both of these approaches have disadvantages.

Modifying the MOS requires mapping flight capabilities onto ground representations which may not adequately implement them, causing flight capabilities to be inaccessible to or poorly modeled for the operations team. Modifying the flight software may prove an undue burden on personnel, flight resources, and development processes. Differing priorities for costing and implementing the modifications may cause the flight and ground organizations to be at odds. Misunderstanding of functionality may lead to incorrect behavior of the end-to-end system that adds risk and requires time and resources to correct. Accurate schedule estimates for the work may be difficult to derive given the complexity of integration.

Since every spacecraft requires a radio, and since modern software-defined radios using the STRS standard can host complex applications, a third option becomes possible: host the command and sequencing portion of the flight/ground interface within the radio itself. This leaves the host flight software and the ground data system unchanged, allows personnel to use a familiar sequencing capability, and requires little additional investment of time and money outside of defining mission-specific commands. In addition, since radios are frequently furnished by JPL to its vendor-built spacecraft, a simple means for integrating time-tested heritage sequence capabilities is provided by simply delivering a radio that already includes the desired software.

A. Radio-hosted sequencing

Figure 6 shows a flight software core and VML flight software integrated with an STRS implementation as it would be hosted in a suitable software-defined space radio. Two such radios have been developed by JPL: the Universal Space Transponder [18], and the Iris radio [19]. Other STRS-compatible radios should reach the market in the coming years, and could accommodate standardized flight / ground interface elements given sufficient unused resources, including CPU cycles, memory, and file system space. The flight software core could be any of a number of different available systems, including Core Flight System / Core Flight Executive [17] (an open source spacecraft flight software package provided by NASA) and VML 3 (provided by BSE). While STRS calls out RTEMS, it may also be possible to adapt the system to work with other real-time operating systems such as VxWorks. Hosting the command elements of the flight / ground interface in the radio could therefore allow a great deal of flexibility in selection of the software components to be hosted.

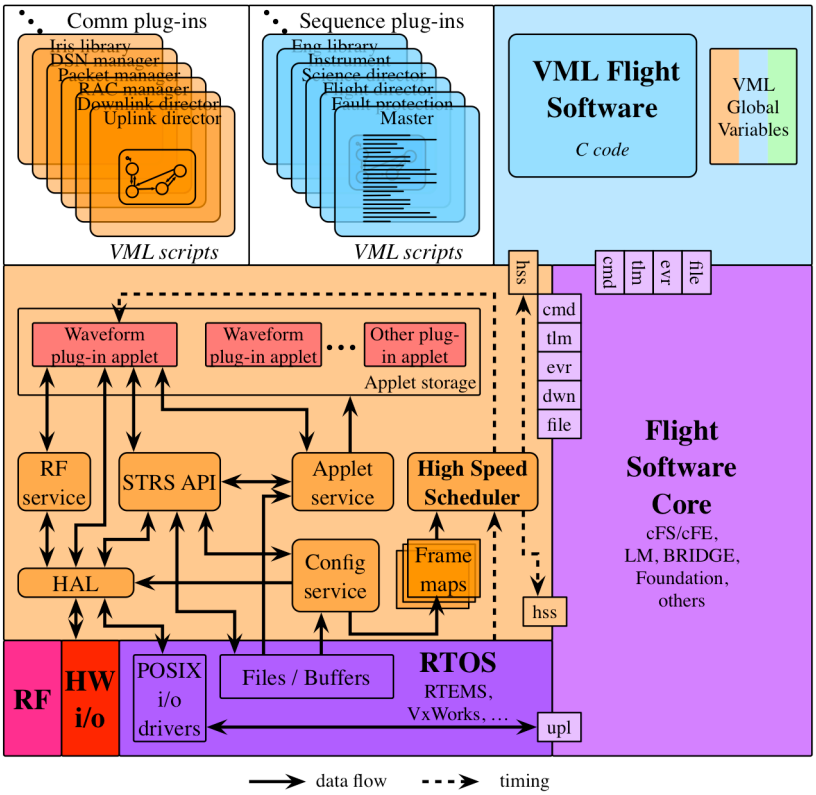

Figure 6: STRS hosting flight software core and VML 3

The resulting command flow shown in Figure 7 ends up looking very much like the hardware command decoder of early space missions shown in Figure 2. Commands arrive onboard the spacecraft from the ground system. Immediate commands for the radio are interpreted within the radio software elements, and those intended for the host spacecraft are passed through to the CDH for interpretation. Sequences dispatch commands which are either routed to software elements in the radio or out to the CDH. Also shown is a return path of telemetry from the CDH for downlink, which also may be used to determine conditions governing logic within sequences.

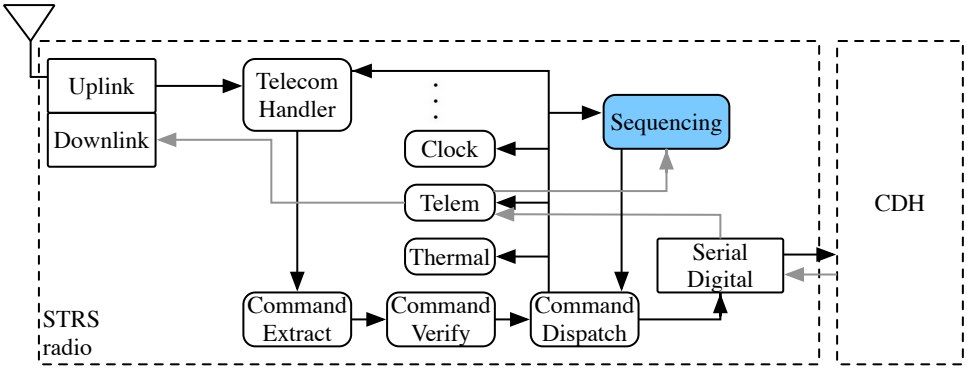

Figure 7: Command and telemetry flow of radio-hosted FGI command element

With the command and sequencing element of the flight / ground interface in the radio, the overall system appears as in Figure 8. Implementation of the flight/ground system no longer requires extensive mission adaptation on the ground side, nor do vendor-provided elements of the flight system have to be altered to accommodate the standardized sequencing system.

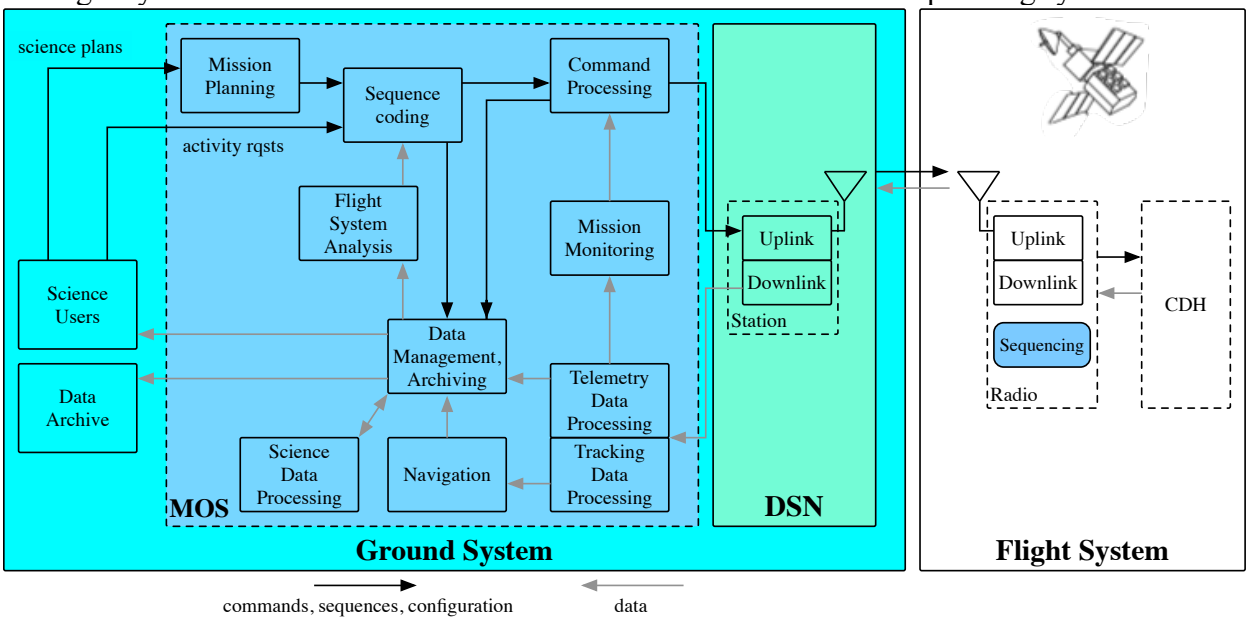

Figure 8: Sequencing element of the flight / ground interface hosted in the radio

B. Radio-hosted service: automated navigation, trajectory calculation, and maneuver derivation

In addition to sequencing, other services could be hosted in the radio. Once a flight software core is in place and using the STRS software infrastructure, the ability to interface to various services is greatly simplified. One such service is the AutoNav[7][8] software, built by JPL to perform on-board optical orbit determination, trajectory calculation, and maneuver derivation.

This sophisticated software could prove challenging to integrate into a vendor's flight software build, but can be easily accommodated within the STRS environment alongside VML and the flight software core. Figure 9 illustrates the inclusion of AutoNav. This type of service could prove to be a powerful incentive to missions to adopt onboard autonomy at low risk and low cost, thereby saving host missions money, decreasing personnel costs, and reducing mission risk.

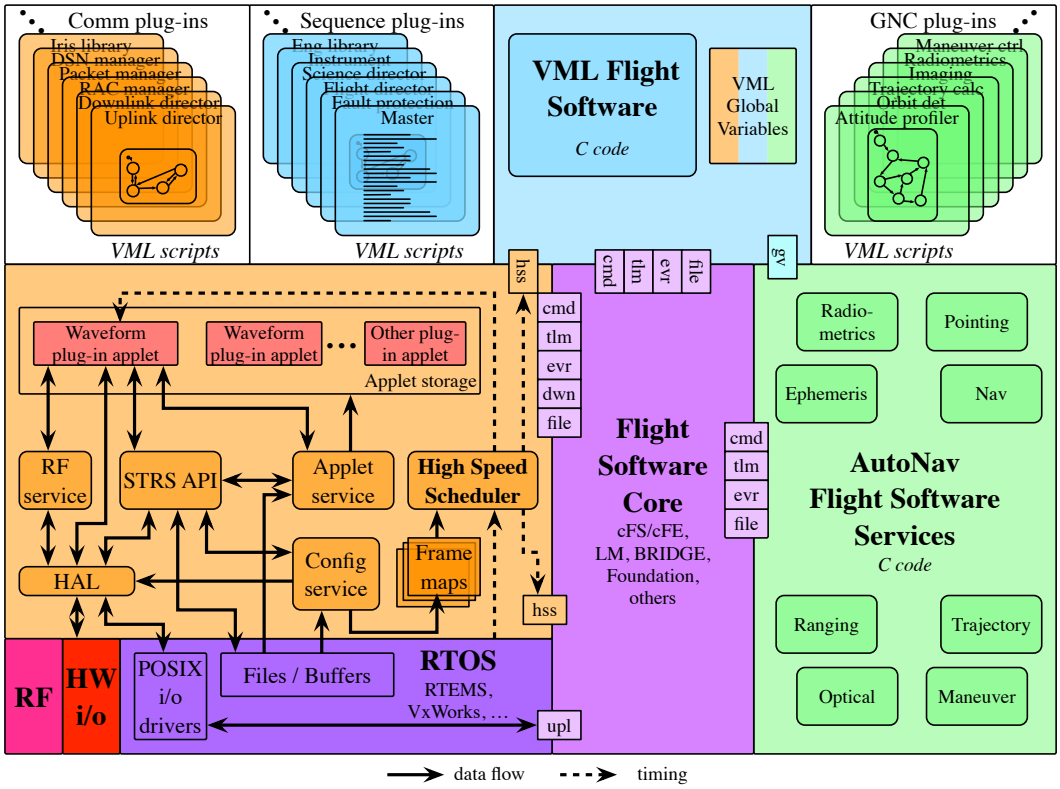

Figure 9: Radio-hosted VML sequencing with AutoNav

IV. Potential host deep-space STRS radios

C. Universal Space Transponder and Iris Radio

Hardware for the Universal Space Transponder (UST) [18] and the Iris radio version 2 [19] appear in Figure 10. These two radios are both built around STRS and form the backbone of future transponder offerings from JPL. The UST is targeted at large spacecraft with high power transmission and high-speed data, and can operate in the UHF-band, S-band, X-band, and Ka-band frequencies, either exclusively or in combinations. The Iris radio is targeted at small spacecraft, including those with a CubeSat form factor, and is currently available in the X-band and UHF-band frequencies. Iris is also commercially produced by the Spacecraft Dynamics Lab, a non-profit research corporation of the Utah State University Research Foundation.

VML sequencing requires approximately 1.5 MB of memory for executable code, data, and sequence files. Various flight software cores and real-time operating systems may range in memory footprint from 2 MB on up. UST and Iris currently offer 384 MB (3 Gbits) and 2 MB of SRAM, respectively. While the current UST hardware should provide enough memory to support both its own internal operation and hosting external software, future expansion of the on-board SRAM in Iris would enable it to serve as a host.

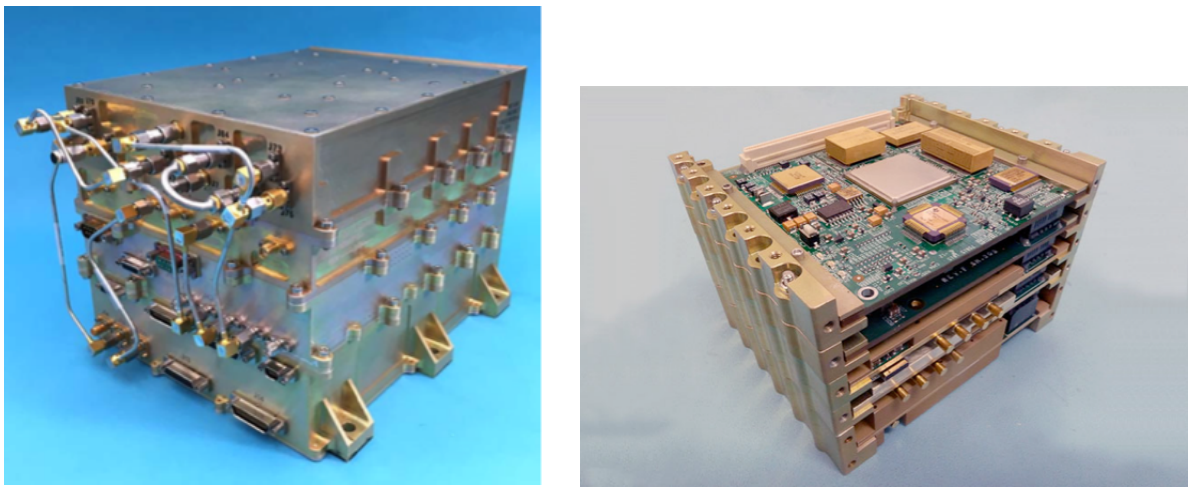

Figure 10: Universal Space Transponder (left) and Iris radio v. 2 (right)

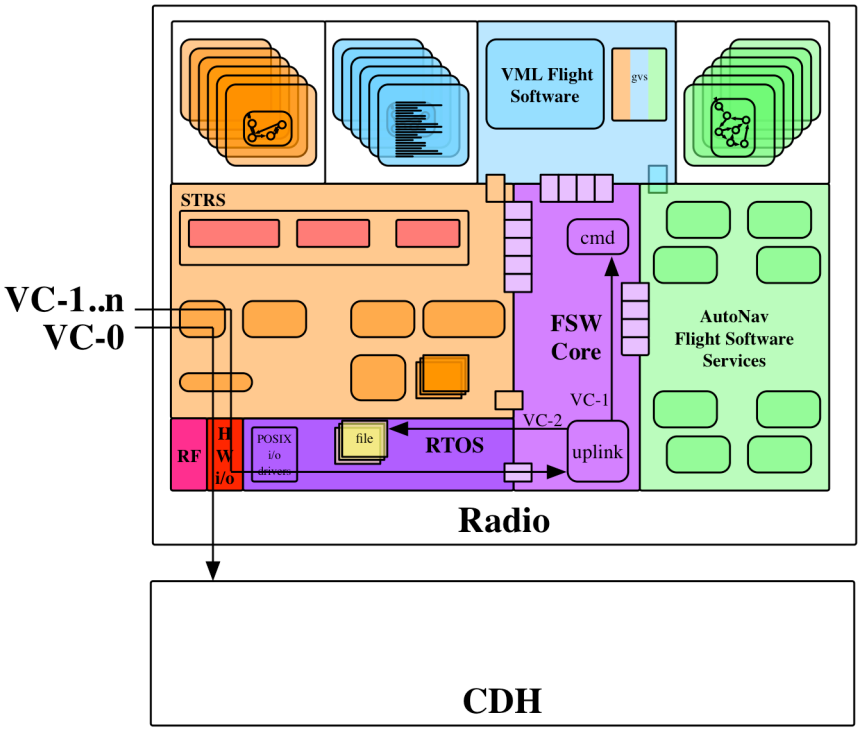

Figure 11: Virtual channels and APIDs route radio commands, files

D. CCSDS Command routing

The CCSDS standard for packetization and framing of spacecraft uplink products contains two routing features that would prove useful for radio-hosted sequencing: the *virtual channel* [20] and the *application process identifier* (APID) [21]. A virtual channel value between 0 and 63 associates an arriving packet with a desired command destination. By convention on many JPL deep space missions, VC-0 is used for hardware commands to the radio, VC-1 provides software commands for immediate execution, and VC-2 handles file uploads. For each virtual channel, an APID value of between 0 and 2039 is available in the packetization scheme to associate arriving command packets with applications that can react to those packets. These conventions are followed in the command routing technique described here. For instance, suppose a spacecraft is not already using VC-1 and VC-2 for its own purposes. Figure 11 shows the arrival of packets on various virtual channels, with VC-0 packets resulting in setting of a discrete signal, VC-1 being forwarded to the internal command dispatcher within the radio, and VC-2 resulting in the deposition of files within the file system.

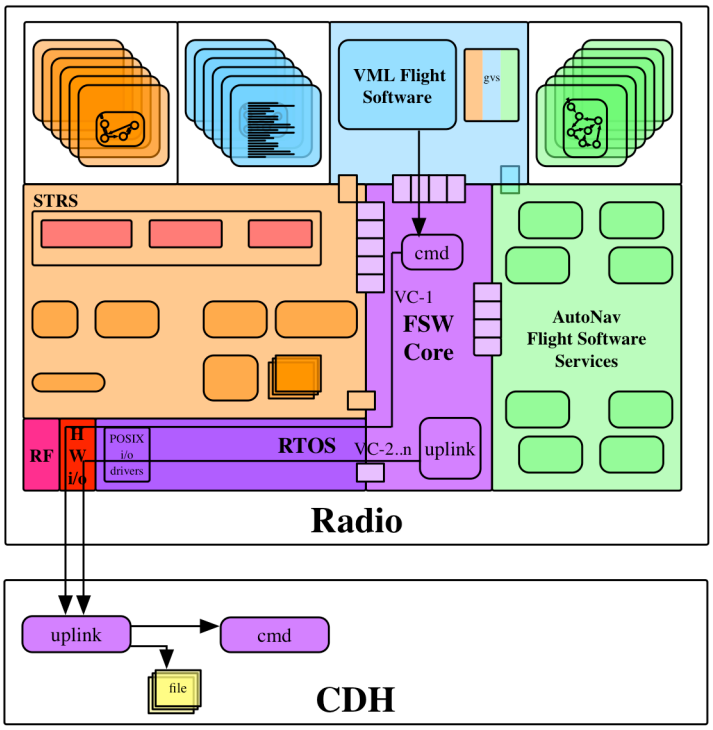

Figure 12: Forwarding of spacecraft command and file data to CDH

Each vendor spacecraft and associated instrumentation will have its own specification of VC and APID usage, requiring the radio usage of APIDs and spacecraft / instrument usage of VCs and APIDs be deconflicted. In cases where virtual channels are used in common between the radio and the CDH, the radio can be programmed to use specific APIDs that are not in use on the host CDH for its commands and file data on VC-1 and VC-2.

Note that spacecraft commands implemented using opcode definitions provide another avenue to differentiate between radio commands and spacecraft commands. The radio command opcodes are chosen to deconflict them from the spacecraft commands, regardless of the APID and virtual channel on which the commands are delivered. Doing so requires the radio to examine each immediate command upon arrival or issuance from sequencing to make the evaluate / forward decision.

Forwarding of command and file data from the radio into the spacecraft CDH is illustrated in Figure 12. Commands on VC-1 which are not intended for the radio are identified using APID or opcode values and forwarded out the serial link to the CDH as CCSDS packets. In a similar fashion, file data on VC-2, and all data on VC-3 through VC-63 is forward to the CDH.

E. Telemetry reporting

Downlink from the spacecraft includes telemetry produced by the CDH and telemetry produced by the operation of the radio and its applications. The radio can be configured to use a technique known as *prechannelization*, in which the format of a telemetry packet is variable, consisting of pairs of identifiers and values that are interpreted after receipt on the ground in order to extract telemetry. APID usage between the CDH and the radio can be easily deconflicted by reserving certain APIDs which are not used by the CDH as originating from the radio.

It may be necessary to interpret a subset of the telemetry originating from the CDH in order to allow sequences running in the radio to react to spacecraft conditions. VML 3 allows reactive responses based on spacecraft conditions and keeps a mirror of reported telemetry items in its global variable list. The data flow of these items is illustrated in Figure 13. In cases where the overhead associated with extracting and interpreting telemetry items from every received CCSDS packets proves impractical due to limitations on available CPU and i/o resources, a specific report with a limited number of measurements may be sent to the radio from the CDH using a designated APID.

V. Advantages of standardized radio-hosted flight/ground interface

Every flight system has some kind of interface with the ground. As most deep-space missions are unique, parts of their flight software may also be unique. Without a standardized interface, each flight system will need to have a customized interface.

For most missions, there is an existing ground software/hardware system that will be adapted to command and monitor the mission. In most of these ground systems, the sometimes-painful lessons learned from years of spacecraft operations have been built into software, processes, and procedures. This "corporate memory" is invaluable and must be maintained. While the ground system often evolves over many missions, either each new spacecraft design will entail changing the ground system, or each spacecraft must be designed to interface with the ground system as it stands. Systems that support a number of different missions concurrently can become inflexible, and changing them imposes risk.

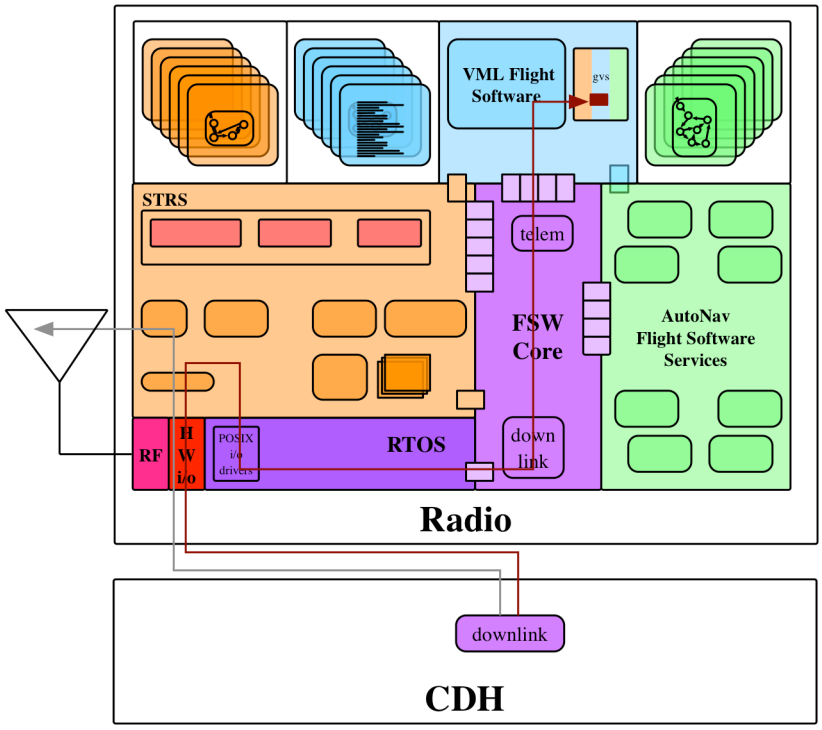

Figure 13: CDH sending normal telemetry to ground and engineering subset to sequencing

This situation requires mission managers to have to make a difficult choice between changing the ground to match flight (slow and risky, possible risk to multiple missions), changing flight to match the ground (expensive and risky), or some combination of both. This often limits mission developers to only a few vendors with existing, well-understood interfaces. Migrating part of the flight-ground interface onboard puts much-needed flexibility back into the hands of the developers and operators.

A. Lower cost to adapt to new mission

Standardizing how the FGI is adapted to each new mission provides savings in both cost and risk. Much of the information needed by both flight and ground can be pre-planned and pre-formatted, making a first-cut adaptation quick to create and inexpensive to implement. For example, command formats and a map of needed engineering telemetry could be pre-defined, with a map of APIDs created to match standard telemetry types and APID deconfliction routines scripted. A first cut adaptation for testbed use could be made in a matter of hours once some limited spacecraft commands and telemetry are provided. Adjustments during integration and test can easily be made by adding new commands and telemetry items incrementally to the formats in use. From one mission to the next, a "library" of mission adaptations accumulates, and the non-recurring engineering cost virtually disappears.

Using VML as the flight-ground interface system also allows development of mission-independent MOS functions and ground system components, with low-cost adaptations for new

host spacecraft and payloads, and low-cost inclusion of autonomy features, even in flight. Moving the sequencing function from the CDH to the radio allows more straightforward adaptation to each mission. Rather than customizing to each spacecraft's CDH flight software, adaptations need only address radio changes since the most recent mission, and the specific set of commands, sequence constructs, and telemetry of the new mission.

Costs for adaptation drop from many work-years of software development and test under the traditional paradigm down to a low-level effort of adaptation management after a brief early period of adaptation definition. Prior missions provide typical examples for costing a traditionally adapted system. Assuming a mission features a file system onboard, between three and five work years is necessary for the uplink adaptation before launch using the traditional development methodology. Generally, the first year of cruise operations involves significant maintenance time, with additional maintenance required when science operations start. In contrast, embedding the FGI in the radio brings with it a standardized and tested software service infrastructure, and as such would require a fraction of a work year for a new mission. This creates a savings per mission of approximately six work years or more.

B. Lower learning curve for personnel

Standardized interfaces allow for standardized tools, training, and sharing of both development and operations personnel. Sharing of personnel for short-term "surges" in team size during parts of integration and test or launch/checkout is more manageable when shared staff are already fully versed in the tools and processes. Training of new staff speeds up since this training can occur in a range of venues, having concepts shared among the missions. The focus of training for a mission becomes learning the differences among missions rather than first struggling with different concepts, tools, and processes.

C. Reduced schedule risk for adaptation

With a standardized interface pre-defined by the selection of a radio hosting VML, a specific mission adaptation can be defined quickly and safely. Well before any unique flight software is ready for testing, the flight-ground interface can be up and running in a testbed. This allows very early definition and testing of both the radio's interface with spacecraft flight software using a simulation and the adaptation for the mission specifics. This also leads to the ability to simulate operations strategies while changes can still be made to the flight system design. [14]

D. Lower barrier to entry for LEO and GEO providers to deep space

Past deep space missions have relied on highly capable and diversified spacecraft vendors to supply host spacecraft in support of challenging mission environments and specialized payloads. In response, vendors have built complex and customized flight software systems, and the MOS developers have created many custom features to interface with the missions. This has led to unique integration and test environments, unique ground systems, and highly specialized operations procedures. Few vendors have the capabilities or the business models to compete in this type of environment.

By locating the FGI in the deep space radio, virtually any spacecraft vendor can provide spacecraft and participate in deep space missions. The only changes needed are a physical and data connection to the deep space radio and whatever modifications would otherwise be needed for the mission environment. Because of the availability of the UST and Iris deep space radios, many kinds of spacecraft can be included in deep space missions without extensive and expensive flight software modifications. These range among low-cost LEO spacecraft, more

capable GEO spacecraft, and high cost / high complexity spacecraft intended for other types of orbits such as for outer planets missions. Many vendors and scientific organizations lack an in-house flight software customization capability: these vendors would be able to participate in deep space missions without the need for significant flight software changes.

Organizations providing MOS services will also have lower barriers to participation. The standardized FGI in the radios allows their services to be independent of the flight software interfaces of the host spacecraft. Diverse and distributed operations organizations could then interact with a wide variety of spacecraft providers without having to develop a unique infrastructure. This has been demonstrated on JPL missions where instrument payload organizations successfully operate those instruments using their in-house operations systems and teams, maintaining independence from the host spacecraft and the JPL MOS.

Mission development organizations, formerly exclusively government entities, are growing in diversity as well. Private groups, commercial companies, educational organizations, and consortia of any of these are now planning Earth orbiting missions and could soon attempt deep space missions. The ability to select spacecraft vendors, operations system providers, and payload providers from a broad and diverse pool *and connect them inexpensively* through a standardized flight/ground interface can enable such missions by keeping costs reasonable.

VI. Conclusions

The growing use of software defined radios, including the Iris and the Universal Space Transponder from JPL, provides a unique opportunity to simplify the flight/ground interface and standardize deep space operations. The presence of STRS within these radios furnishes an application hosting capability which can run complex sequencing software such as VML 3, and specialized navigation services including AutoNav. These capabilities would allow considerable automation standardization for a wide variety of deep-space missions, and standardize the flight/ground interface of these missions. Standardizing the FGI in this way yields cost and risk reductions, and presents a simple and convenient means for providing spacecraft which were originally designed for LEO and GEO with the operations capabilities and autonomy needed for accomplishing challenging deep space missions.

Acknowledgments

Some of the work described in this paper was carried out by Blue Sun Enterprises, Inc., under an agreement with the National Aeronautics and Space Administration, and administered by the Office of Chief Technologist as a Small Business Innovation Research grant. Part of this work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

Reports, Theses, and Individual Papers

[1]Grasso, C. A., Lock, P. d., "VML Sequencing: Growing Capabilities over Multiple Missions", AIAA Space Operations Conference Proceedings, April 2008.

[2]Grasso, C. A., "The Fully Programmable Spacecraft: Procedural Sequencing for JPL Deep Space Missions Using VML (Virtual Machine Language)", IEEE Aerospace Applications Conference Proceedings, March 2002.

[3]Grasso, C. A., "Techniques for Simplifying Operations Using VML (Virtual Machine Language) Sequencing on Mars Odyssey and SIRTF", IEEE Aerospace Applications Conference Proceedings, March 2003.

[4]Peer, S. and Grasso, C. A., "Spitzer Space Telescope Use of Virtual Machine Language", IEEE Aerospace Conference Proceedings, December 2004.

[5]Grasso, C. A., "Virtual Machine Language (VML)", NPO 40365, JPL Commercial Programs Office, Innovative Technology Asset Management Group, Docket Date: 12-May-2003.

[6]Grasso, C. A., "Virtual Machine Language (VML) NASA Board Award", NASA Inventions and Contributions Board, NASA Technical Report 40365, Award Date: September 7, 2006.

[7]Riedel, J. E., et. al., "AutoNav Mark 3: Engineering the Next Generation of Autonomous Onboard Navigation and Guidance", AIAA Guidance, Navigation, and Control Conference, August 2006.

[8]Riedel, J. E., Grasso, C. A., et. al., "Configuring the Deep Impact AutoNav System for Lunar, Comet and Mars Landing", AIAA Astrodynamics Specialist Conference, August 2008.

[9]Grasso, C. A., Riedel, J. E., "VML 3.0 Reactive Sequencing Objects and Matrix Math Operations for Attitude Profiling", AIAA Space Operations Conference Proceedings, May 2012.

[10]Grover, M., Cichy, D., Dasai, P.N., "Overview of the Phoenix Entry, Descent and Landing System Architecture," AIAA Paper AIAA 2006-7218, AIAA/AAS Astrodynamics Specialist Conference; Honolulu, HI, 18-21 August 2008.

[11]Garcia, M., Fujii, K., "Mission Design Overview for the Phoenix Mars Scout Mission," AAS Paper 07-247, AIAA/AAS Space Flight Mechanics Meeting; Sedona, AZ, 28 January -01 February 2007.

[12]Grasso, C. A., Riedel, J. E., Vaughn, A.T., "Reactive Sequencing for Autonomous Navigation Evolving from Phoenix Entry, Descent, and Landing", AIAA Space Operations Conference Proceedings, April 2010.

[13]Wertz, J. R., Larson, W. J, Space Mission Analysis and Design, 3rd edition, (c)1999, W. J. Larson and Microcosm, Inc., pp. 397-398.

[14]Grasso, C. A., Lock, P. d., "Flight-Ground Integration: the Future of Operability", Space Operations: Innovations, Inventions, and Discoveries, edited by Cruzen et al., 2015.

[15]Space Telecommunications Radio System (STRS) Architecture Standard, NASA Technical Standard Rationale NASA-HDBK-4009, Approved: 06-05-2014.

[16]Real-Time Executive for Multiprocessor Systems (RTEMS) home page, www.rtems.org.

[17]cFS/cFE, Core Flight System / core Flight Executive home page, cfs.gsfc.nasa.gov/Introduction.html.

[18]Pugh, M., et al., "The Universal Space Transponder: A Next Generation Software Defined Radio", IEEE Aerospace Conference Proceedings, 2017.

[19] Duncan, C., et al. "Iris Transponder – Communications and Navigation for Deep Space," Proceeding of the 28th Annual AIAA/USU Conference on Small Satellites, 2014.

[20]Consultative Committee for Space Data Systems Recommendation for Space Data System Standards, Telecommand Part 2, Data Routing Service, Blue Book, CCSDS 202.0-B-3, June 2001.

[21]Consultative Committee for Space Data Systems Recommendation for Space Data System Standards, Space Link Identifiers, Recommended Standard, Blue Book, CCSDS 135.0-B-3, October 2006.

Related web sites

Blue Sun Enterprises VML Website	www.bluesunenterprises.com
Space Dynamics Laboratory Website	www.spacedynamics.org